OULUN SEUDUN
AMMATTIKORKEAKOULU

Veli-Ilari Kuure

BBIC-EMULAATTORIN OHJAUSRAJAPINNAN TOTEUTUS RFIC-PIIRIN TESTAUSJÄRJESTELMÄÄN

BBIC-EMULAATTORIN OHJAUSRAJAPINNAN TOTEUTUS RFIC-PIIRIN TESTAUSJÄRJESTELMÄÄN

Veli-Ilari Kuure
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, langattomat laitteet

Tekijä: Veli-Ilari Kuure
Opinnäytetyön nimi: BBIC-emulaattorin ohjausrajapinnan toteutus RFIC-piirin testausjärjestelmään
Työn ohjaajat: Rami Määttä (Renesas Mobile Europe Oy), Veijo Korhonen (OAMK)
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 34

Opinnäytetyössä toteutettiin BBIC-emulaattorille ohjausrajapinta RFIC-piirin testausjärjestelmään. Tärkeimpänä tavoitteena oli saada ohjausrajapinta integroitua sujuvasti testausjärjestelmään ja toimimaan testausjärjestelmän vaatimusten mukaisesti. Toiseksi tavoitteeksi asetettiin itsenäisen ohjauskäyttöliittymän toteuttaminen. Työn tilaajana toimi Renesas Mobile Europe Oy.

Ohjausrajapinta ja itsenäinen ohjauskäyttöliittymä toteutettiin LabVIEW'lla. Ohjausrajapinnan muutamassa osiossa käytettiin apuna C++-koodikielellä toteutettuja ohjelmia. Työssä tuli myös osata käyttää joitakin mittalaitteita testausvaiheessa.

Ohjausrajapinnan integrointi onnistui tavoitteiden mukaisesti. Ohjausrajapinnan ansiosta RFIC-piirin testausta voidaan suorittaa useammalla testipaikalla, koska Renesasin kehittämiä BBIC-emulaattoreita on laajemmin saatavilla. Lisäksi integroitu ohjausrajapinta osoittauti nopeammaksi kuin aikaisemmin käytetyn BBIC-emulaattorin ohjausrajapinta. Toinen tavoite onnistuttiin saavuttamaan hyvin ja ohjauskäyttöliittymä saatiin toteutettua toimivaksi kokonaisuudeksi. Itsenäinen ohjauskäyttöliittymä tuo helpotusta pienempien yksittäisten testauksien suorittamiseen ja RFIC-piirillä ilmentyvien ongelmatilanteiden selvittämiseen.

Asiasanat: RFIC, BBIC, LabVIEW, testausjärjestelmä

ABSTRACT

Oulu University of Applied Sciences
Information Technology and Telecommunications, Wireless Devices

---

Author: Veli-Ilari Kuure
Title of thesis: BBIC emulator control interface for the RFIC test system
Supervisors: Rami Määttä (Renesas Mobile Europe Oy), Veijo Korhonen (OUAS)
Term and year when the thesis was submitted: Spring 2013 Pages: 34

The purpose of this thesis was to implement the BBIC emulator control interface for the RFIC test system. The most important goal for the thesis was to integrate control interface successfully into the RFIC test system. The second goal for the thesis was to implement a standalone control interface which is independent from the RFIC test system. The thesis was commissioned by Renesas Mobile Europe Oy.

Both of the control interfaces were implemented in the LabVIEW programming environment. In some sections of the BBIC emulator control interface the C++ programming language was used to assist some subprograms. In addition, it was required to be able to use some measurement equipment for testing the control interface.

The BBIC emulator control interface was successfully integrated and met all the requirements. Because of the successful integration RFIC testing can now be done in several places at the same time. Furthermore, the integrated control interface turn out to be faster than the previous control interface. The second goal was also successfully reached and the standalone control interface was implemented as a fully functioning program. The standalone control interface makes it easier to test small individual cases and to debug errors occurred in the RFIC.

Keywords: RFIC, BBIC, LabVIEW, Test system

ALKULAUSE

Tämä opinnäytetyö on tehty kevään 2013 aikana Renesas Mobile Europe Oy:n tiloissa. Ennen opinnäytetyön aloittamista vuoden 2013 alussa olin työskennellyt Renesasilla vuoden 2012 kesäkuusta alkaen.

Ensiksi haluan kiittää kaikkia RFIC Test Teamin jäseniä työssä opastuksesta ja rennosta työilmapiiristä koko Renesasilla työskentelyni ajalta. Erityisesti haluan kiittää Mikko Savolaa opinnäytetyöpaikan järjestämisestä ja Rami Määttää opinnäytetyöhön antamastaan ohjauksesta. Kiitokset myös OAMK:n puolen työn ohjaajalle Veijo Korhoselle.

Oulu 23.5.2013

Veli-Ilari Kuure

SISÄLLYS

SANASTO

BBIC	Baseband Integrated Circuit eli integroitu baseband-piiri
FPGA	Field-Programmable Gate Array on ohjelmoitava digitaalinen logiikkapiiri.
I/Q-data	Ilmaisee vaihtelun signaalin voimakkuudessa ja vaiheessa. Käytetään signaalin moduloinnissa.
RFIC	Radio Frequency Integrated Circuit eli integroitu radiotaajuuspiiri
RX	Telekommunikaatiossa käytetty lyhenne vastaanottimelle.
TX	Telekommunikaatiossa käytetty lyhenne lähettimelle.
VI	Virtual Instrument on LabVIEW’n tuottama graafinen ohjelma.
VSA	Vektorisignaalianalysaattori

1 JOHDANTO

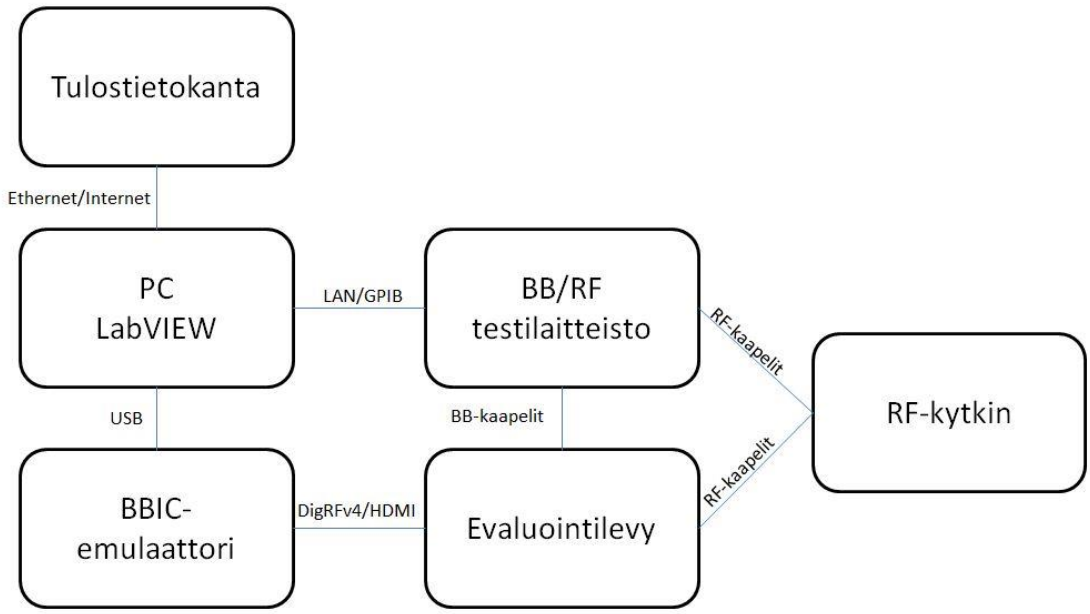
Tämän päivän RFIC-piirit kattavat monia eri standardeja ja taajuuksia. Näin ollen RFIC-piirin testauksessa joudutaan ajamaan suuri määrä erilaisia testejä. Kustannustehokkaassa toiminnassa aika ei enää riitä testaamaan piirejä manuaalisesti, joten on jouduttu kehittämään monimutkaisia ja laajoja automaattisia testausjärjestelmiä. Myös työn teettäjä Renesas Mobile Europe Oy kehittää omaa automaattista testausjärjestelmää vanhan tilalle. Kehitteillä oleva järjestelmä tulee olemaan monipuolisempi, nopeampi ja helppokäyttöisempi kuin entinen testausjärjestelmä.

Opinnäytetyö on osa tätä kehitteillä olevaa automaattista testausjärjestelmää. Työn tarkoituksena oli toteuttaa testausjärjestelmään uusi ohjausrajapinta, joka mahdollistaa Renesasin oman BBIC-emulaattorin käytön RFIC-piirien testauksessa. Aikaisemmin testausjärjestelmässä RFIC-piirin DigRF-rajapinnan ohjaus oli mahdollista ainoastaan erään mittalaitevalmistajan kehittämällä BBIC-emulaattorilla. Näitä BBIC-emulaattoreita Renesasin toimipaikoilla on kuitenkin vähän, joten Renesasin omalle BBIC-emulaattorille haluttiin tuki testauksen nopeuttamiseksi ja monipuolistamiseksi.

Työn tärkeimpänä ja ensimmäisenä tavoitteena oli toteuttaa ja integroida RFIC-piirin ohjaus kehitteillä olevaan automaattiseen testausjärjestelmään. Toisena tavoitteena oli tehdä Renesasin omalle BBIC-emulaattorille itsenäinen ohjaustyökalu, joka on täysin riippumaton automaattisesta testausjärjestelmästä. Suurimman haasteen RFIC-piirin ohjaukselle asettaa automaattisen testausjärjestelmän vaatima nopeus. Ohjauksen tavoitteena oli suoriutua tehtävistään maksimissaan sekunnissa, joka osoittautui erittäin haasteelliseksi toteuttaa.

2 RFIC-PIIRIN TESTAUSJÄRJESTELMÄ

Renesasin kehittämä testausjärjestelmä vastaa hyvin tämän päivän testauksen vaatimuksia. Automaattinen testausjärjestelmä pystyy ajamaan RFIC-piirin karakterisoinnissa käytettyjä RF- ja BB-testejä. Järjestelmän suurimpana hyötynä on sen kyky ajaa kaikki karakterisointiin käytettävät testitapaukset, kun RFIC-piiri on kiinnitetty järjestelmään. Eri mittausasetuksia saadaan muodostettua käyttämällä kauko-ohjattuja DC- ja RF-kytkimiä. Testausjärjestelmä koostuu pääpiirteittäin testausohjelmaa ajavasta PC:stä, BB- ja RF-testilaitteista, BBIC-emulaattorista ja evaluointilevystä. (Kuva 1.)



KUVA 1. Kuvaus testijärjestelmän tärkeimmistä kokonaisuuksista

Kuvassa näkyvät automaattisen testausjärjestelmän tärkeimmät osiot ja niiden väliset yhteydet. Monipuolisen testauksen mahdollistamiseksi testausjärjestelmästä löytyy tuki muutamalle erilaiselle lämpötilan hallintalaitteelle, kuten lämpökaapille.

Automaattisen testausjärjestelmän ytimenä toimii LabVIEW’lla toteutettua testausohjelmaa suorittava tietokone. Käyttäjälle avautuvasta testiohjelman graafisesta käyttöliittymästä voidaan valita haluttu testitapaus tai rakentaa uusi

testijono. Valinnan jälkeen mittaus käynnistetään, jolloin ohjelma alkaa alustaa testissä käytettäviä mittalaitteita. Kun kaikki mittalaitteet on alustettu testiä varten, testausjärjestelmä antaa BBIC-emulaattorin kautta RFIC-piirille ohjauskomennot. Tämän jälkeen varsinainen testimittaus aloitetaan, ja mittauksen valmistuttua testiohjelma saa tulokset joko BB/RF-testilaitteistolta tai BBIC-emulaattorilta saadulle datalle tehdystä analyysista.

Testitapaukselle määritettyjen asetusten mukaan testausohjelma voi tehdä useita iteraatioita esimerkiksi eri RFIC-piirin asetuksilla. Ohjelmalla voidaan myös ajaa useampia eri testitapauksia peräkkäin ilman käyttäjältä vaadittavia toimenpiteitä. Testitulokset ohjelma tallentaa käyttäjän valitsemiin ohjelman tukemiin tiedostoihin, joita voidaan analysoida manuaalisesti tai haluttaessa ladata tulostietokantaan.

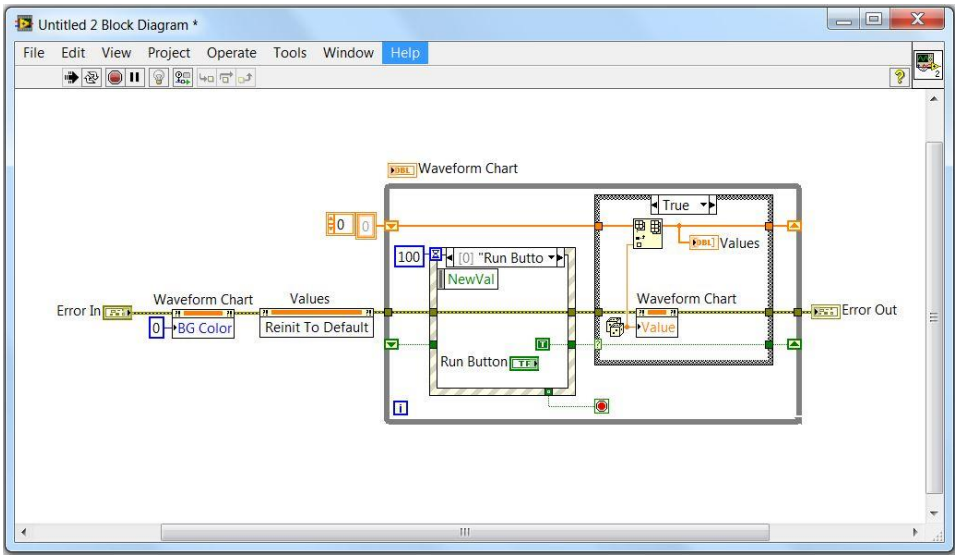
3 KÄYTETYT MENETELMÄT JA LAITTEET

3.1 Ohjelmistoympäristö

3.1.1 LabVIEW

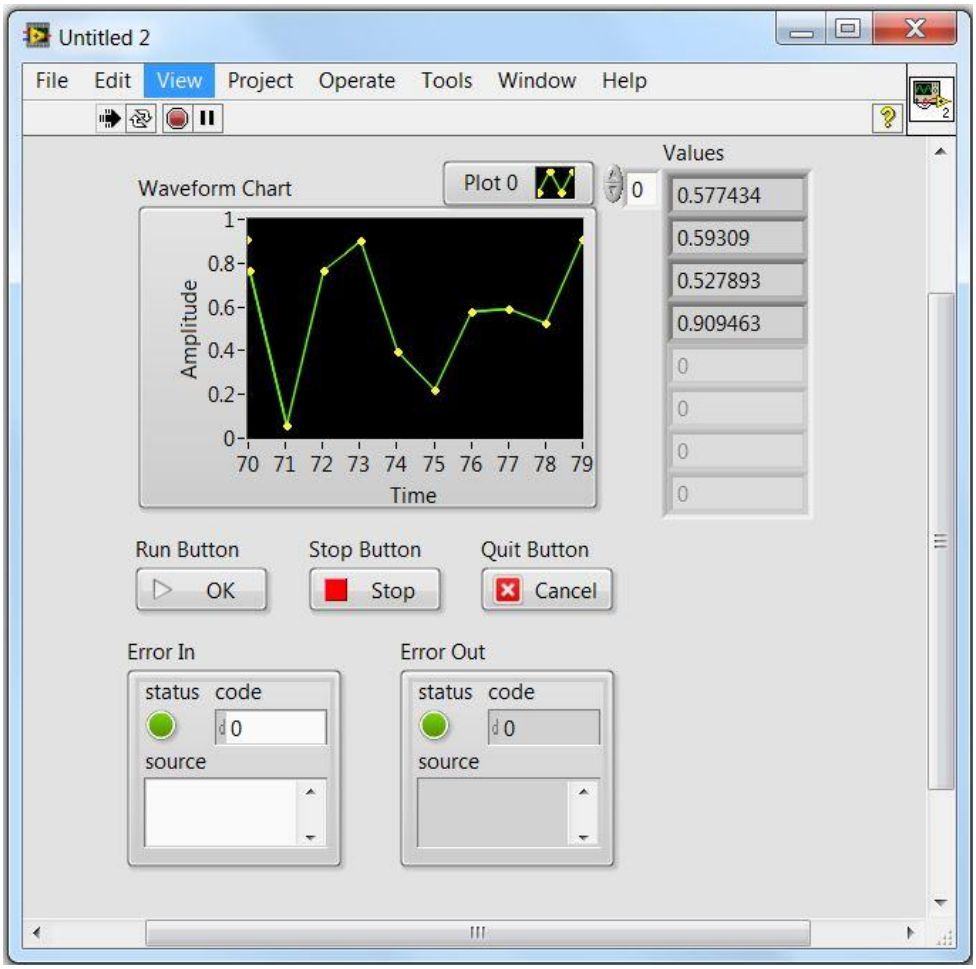
LabVIEW (Laboratory Virtual Instrument Engineering Workbench) on National Instrumentsin kehittämä graafinen ohjelmointiympäristö, jonka ensimmäinen versio julkaistiin vuonna 1986 Macintoshille. Ohjelmalla voidaan helposti ja havainnollisesti toteuttaa monipuolisia ohjelmia, kuten automaattiset testi- ja validointijärjestelmät, mittalaitteiden hallintajärjestelmät ja erilaiset monitorointijärjestelmät. (1)

LabVIEW’n tuottamia graafisia ohjelmia kutsutaan Virtual Instrumenteiksi eli VI:ksi. VI sisältää lohkokaavion (Block Diagram) ja etupaneelin (Front Panel). Lohkokaavio sisältää graafisen koodin, jonka LabVIEW toteuttaa dataflow-periaatteen mukaisesti. Kuvassa 2 on esimerkki LabVIEW’n graafisesta koodista. Dataflow-periaatteella tarkoitetaan koodin suorittamista vaiheittain eli jokainen koodin komponentti suoritetaan sitten kun kaikki komponentin sisääntulot ovat saaneet vaadittavat tiedot. Kun komponentti on suorittanut omat toimintonsa, tulos saadaan komponentin ulostulosta seuraavalle komponentille. (2, s. 5)



KUVA 2. Esimerkki LabVIEW’n lohkokaaviosta

VI:n etupaneeli toimii tehdyn koodin graafisena käyttöliittymänä. Se sisältää kaikki kontrollikomponentit, jotka on luotu lohkokaavioon. Etupaneelissa käyttäjä voi syöttää dataa erilaisiin kontrolleihin, joista data välittyy lohkokaavioon. Kuvassa 3 on esimerkki LabVIEW'n etupaneelista.



KUVA 3. Esimerkki LabVIEW'n etupaneelista

LabVIEW 2012 -ohjelma oli tässä työssä pääasiallisena kehitysympäristönä. LabVIEW'n valinta kehitysympäristöksi oli itsestäänselvyys, koska automaattinen testijärjestelmän toteutus oli tehty LabVIEW'lla.

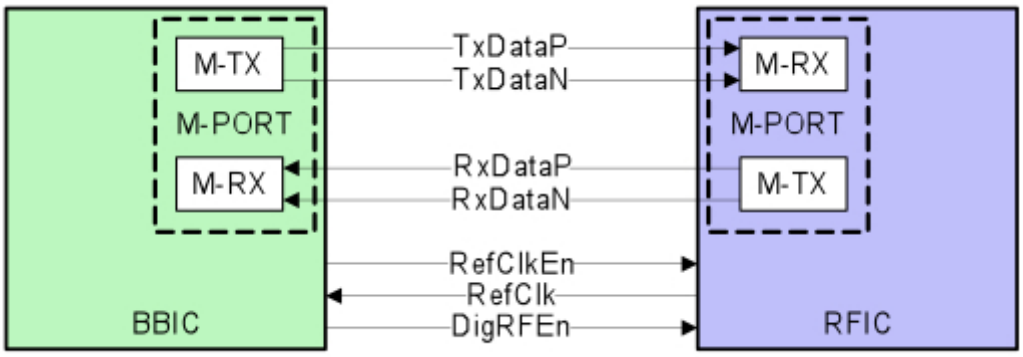
3.1.2 DigRF v4

DigRF v4 on MIPI Alliancessa toimivan The DigRF^{SM} Working Groupin julkaisema spesifikaatio, joka määrittää BBIC-piirin ja RFIC-piirin välisen

rajapinnan. DigRF v4 on suunniteltu tukemaan uusimpia matkapuhelintekniikoita, kuten LTE:tä ja Mobile WiMaxia, mutta myös kaikki aikaisemmat tekniikat on tuettuna tässä spesifikaatiossa. Spesifikaatio määrittää tehokkaan, joustavan ja laajennettavan rajapinnan yhden tai useamman BBIC:n ja RFIC:n välille käyttäen yhtä terminaalia. Tärkeimpinä vaatimuksina DigRF v4 suunnittelussa oli

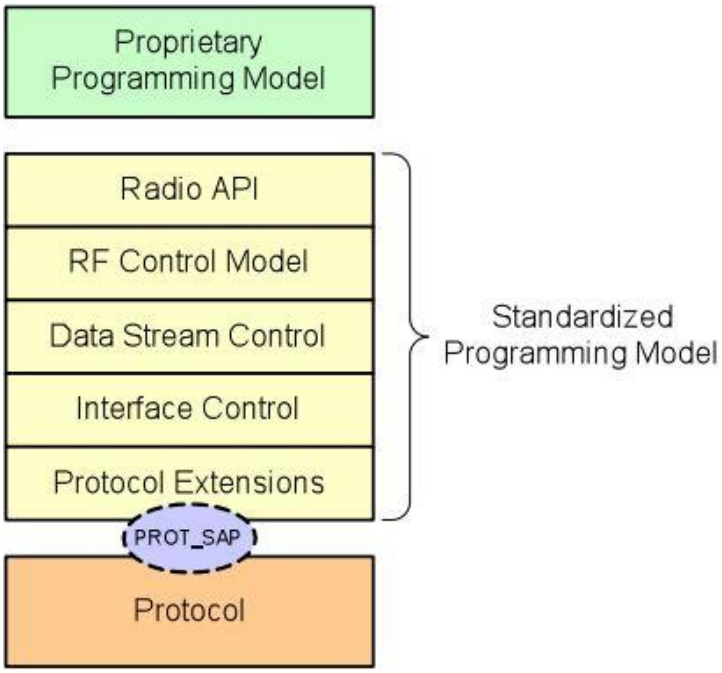
- kasvattaa kaistanleveyttä uuden sukupolven tekniikoiden vaatimuksien mukaisesti
- minimoida rajapinnan pinnien määrä ja rajapinnan kokonaisvirrankulutus
- hallita sähkömagneettisen häiriön aiheuttamia ongelmia välttäen radiotaajuuksille aiheutuvia väärentymiä
- lisätä joustavuutta ja skaalautuvuutta mahdollistaakseen monen vastaanottimen ja lähettimen yhtäaikaisen käytön. (2)

Kuvassa 4 on nähtävissä DigRF v4:n perusasetus ja sen vaatimat eri signaalit. Datalinjoja voidaan halutessa lisätä, mutta perusasetukseen niitä kuuluu vain yksi.



KUVA 4. DigRF-rajapinta (3)

DigRF v4 -spesifikaatio ei määrittele BBIC-piirin tai RFIC-piirin sisäisestä suunnittelusta muuta kuin rajapinnan toiminnan kannalta tärkeimmät yksityiskohdat. Täten piirien eri valmistajien on mahdollisuus toteuttaa kilpailukykyisiä piirejä, kun linjoja ei ole määritetty liian tarkasti. Kuvassa 5 on kuvattu DigRF v4 -spesifikaation ohjelmointirajapintaa. Ohjelmointirajapinta on jaettu standardoituun ja yrityksen omaan ohjelmointimalliin.



KUVA 5. DigRF v4 -standardoidun ohjelmointirajapinnan kuvaus (4)

Kuten kuvasta 5 voidaan nähdä, spesifikaatiossa määritetty ohjelmointitapa rajoittuu muutamalle osa-alueelle protokollan ja patentoidun ohjelmointimallin välille. Näin taataan eri valmistajien piirien toiminta keskenään, mutta kuitenkin ilman suuria rajoituksia yritysten ohjelmointimalleihin.

3.1.3 C++

C++-kieli on kehitetty C-kielen pohjalta monipuolisemmaksi ja tehokkaammaksi kokonaisuudeksi. C++-koodikieleen on lisätty muun muassa luokat ja perinnöllisyys. C++ on yleistynyt monilla eri tuotealueilla, kuten laitteiston kehityksessä ja viihdeohjelmien kehityksessä.

C++-koodikielen osuus työn toteutuksessa jäi hyvin pieneksi, koska sitä käytettiin vain BBIC-emulaattorilta tulevan signaalidatan purkamiseen koodin nopeuden takia. Varsinaiseen RFIC-piirin ohjaukseen C++-kieltä ei tarvinut itse käyttää.

3.1.4 Tulostietokanta

Tulostietokantaa ei varsinaisesti tässä työssä käytetty, mutta se on osa testausjärjestelmän kokonaisuutta, kuten luvussa 2 on kuvattuna. Tietokanta kuuluu automaattisen testausjärjestelmän ohjelmistoympäristöön.

Tulostietokantaa käytetään mittaustulosten tallentamiseen ja niiden analysointiin. Mittaustulokset lisätään tulostietokantaan aina mittauksen jälkeen, mikäli tulokset läpäisevät mittaukselle asetetut vaatimukset. Tulostietokanta mahdollistaa myös tulosten analysoinnin jälkikäteen tietokoneesta riippumatta. Näin ollen itse testiympäristössä käytettävä tietokone on aina vapaana mittauksia varten.

3.2 Laiteympäristö

Tässä luvussa on selitetty tarkemmin laitteista, joita käytettiin apuna ohjausrajapinnan toteutuksessa ja testauksessa. Ohjausrajapinnan toteutukseen tarvittiin vain murto-osaa testausjärjestelmään kuuluvista laitteista.

3.2.1 BBIC-emulaattori

BBIC-emulaattorin tarkoituksena on emuloida RFIC-piiriä ohjaavaa BBIC-piiriä testi- ja evaluointikäytössä. BBIC-emulaattori on tärkeä osa testikokoonpanoa, jotta piiriä voidaan testata mahdollisimman todenmukaisissa olosuhteissa.

Työssä käytetty Renesasin BBIC-emulaattori on yhteydessä tietokoneeseen USB-kaapelilla. BBIC-emulaattorille syötetään tarvittavat tiedot, joista se muodostaa DigRF-spesifikaation mukaisen datakehyksen. Emulaattorille välitettävät tiedot sisältävät datakehyksen kanavatunnuksen, aikaleiman ja varsinaisen datan. Saatuaan nämä tiedot emulaattori lisää datakehykseen virheentarkistusbitit eli niin sanotun CRC-osion. Datakehyksen kanavatunnuksia on kahta erilaista, BBIC-emulaattorin ohjaus ja DigRF-ohjaus. Kanavatunnuksesta riippuen emulaattori suorittaa sille syötetyt ohjauskomennot tai se muodostaa spesifikaation mukaisen datakehyksen. Muodostettuaan DigRF-datakehyksen BBIC-emulaattori välittää sen evaluointilevylle HDMI-kaapelia pitkin.

3.2.2 Evaluointilevy

Evaluointilevyn tarkoituksena on helpottaa RFIC-piirin testausta. Levy tuo testauksessa tarvittavat RFIC-piirin liitännät helposti käyttäjän ulottuville. Testattavana oleva RFIC-piiri asetetaan levylle piiriä varten tehtyyn liitäntään. Liitäntä on suunniteltu siten, että RFIC-piirin rikkoutuessa vaihto onnistuu helposti aukaisemalla liittimen kiinnitykset. Testauksen tärkeimmät liitännät eli RF-liitännät on sijoiteltu mahdollisimman lähelle RFIC-piiriä ylimääräisten häiriöiden välttämiseksi. RF-liitäntöjä on sijoitettu levylle useampia ja ne on jaettu eri taajuuskaistoille siten, että RF-kytkintä käyttäen pystytään ajamaan mahdollisimman monipuolisesti eri testiasetuksia yhdellä testipaikalla.

3.2.3 Signaaligeneraattori

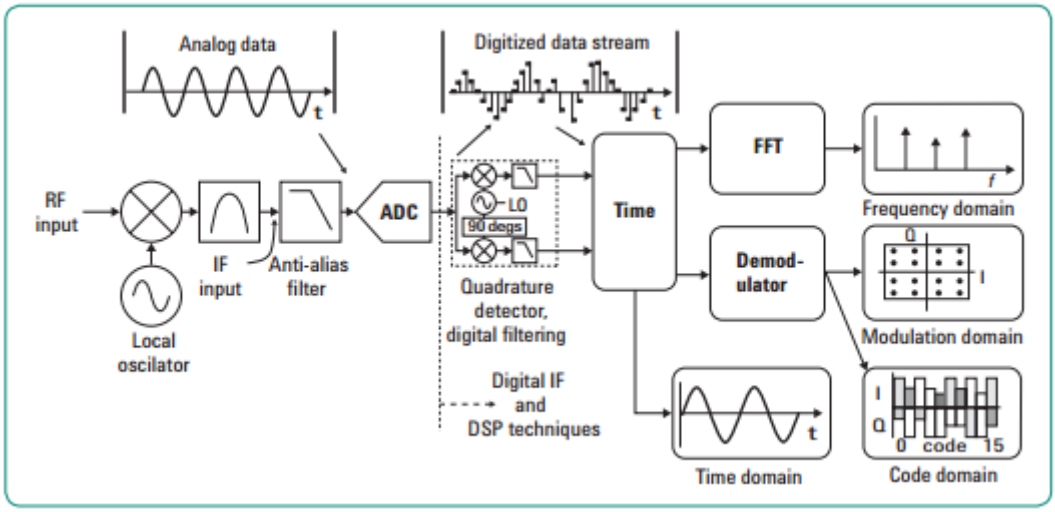
Signaaligeneraattori on yleinen apuväline testauksessa. RFIC-piirin testauksessa sillä voidaan syöttää piirille käyttäjän määrittelemiä erilaisia signaaleja, jotka emuloivat oikeassa käytössä RFIC-piirille tulevia signaaleja.

Testijärjestelmässä käytössä oleviin signaaligeneraattoreihin on mahdollista ladata erilaisten modulaatioiden käyttämiä I/Q-datamalleja. Tällaiset vektorisignaaligeneraattorit ovat edistyksellisempiä malleja yksinkertaisimmista signaaligeneraattoreista. Ainoastaan näillä vektorisignaaligeneraattoreilla saadaan täysin realistinen malli signaalista testejä varten. Kuvassa 6 on esimerkki signaaligeneraattorista.

KUVA 6. Esimerkki signaaligeneraattorista (5)

3.2.4 Vektorisignaalianalysaattori

Vektorisignaalianalysaattorilla eli VSA:lla voidaan tehdä nopeita ja tarkkoja spektrimittausten analysointeja, kuten demoduloinnin analysointi sekä aikatason analysointi. VSA:ta hyödynnetään erityisesti kuvaamaan monimutkaisten signaalien, kuten purskesignaalien ja moduloitujen signaalien laatua. Yksi tärkeimmistä VSA:n ominaisuuksista on pystyä mittaamaan ja manipuloimaan monimutkaista dataa, kuten signaalin vaihe- ja voimakkuustietoja. Kuvassa 4 on esitettynä yksinkertainen kaavio VSA:n toiminnasta. (6, s. 3–4)



KUVA 7. Lohkokaavio VSA:n toiminnasta(6, s. 3)

RFIC-testauksessa TX-mittauksissa VSA:lla mitataan piirin lähettämiä signaaleja ja tutkitaan tuloksista, ovatko piirin tuottamat signaalit vaadittavien standardien mukaisia. RX-mittauksissa piirille saapuneita signaaleja tutkittiin VSA:ta simuloivalla ohjelmalla.

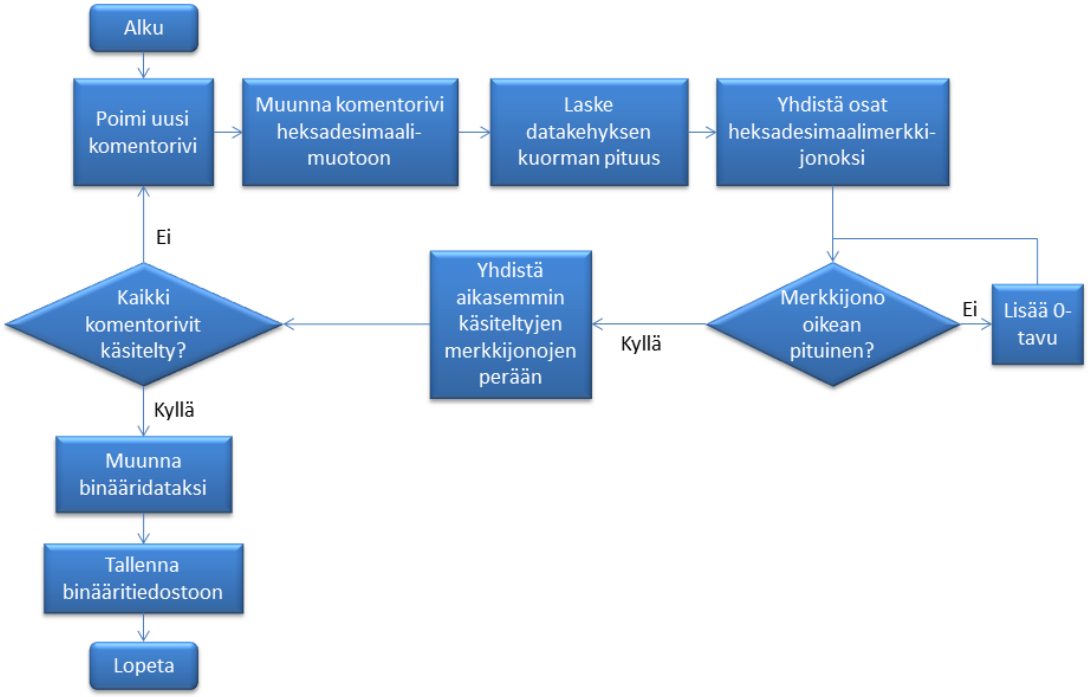
4 TOTEUTUS

Ohjausrajapinnan toteutusvaihe aloitettiin tutustumalla olemassa olevan BBIC-emulaattorin ohjaukseen ja automaattiseen testausjärjestelmään kokonaisuutena. Tutustuminen toiselle BBIC-emulaattorille toteutettuun ohjaukseen antoi osittain kuvan siitä, miten uusi ohjausrajapinta täytyi totetuttaa. Osittaisten yhtäläisyyksien takia ohjausrajapinnasta päädyttiin tekemään yhteensopiva aikaisempaan ohjaukseen tehtyjen komentosanojen kanssa. Näin ollen uuteen ohjausrajapintaan täytyi tehdä aliohjelma, joka osaa kääntää samassa formaatissa olevia komentoja uuden BBIC-emulaattorin vaatimaan binääriformaattiin.

4.1 Binäärikääntäjä

Binäärikääntäjä on koko ohjauksen toiminnan kannalta tärkein osa, sillä tämä osa kääntää kaikki Renesasin BBIC-emulaattorille tehdyt komennot. Kääntäjän sisääntuloon voidaan syöttää yksi tai useampi komentorivi. Komentorivi sisältää aina aikaleiman, kanavatunnuksen ja viimeisenä varsinaisen datan.

Ensiksi kääntäjä poimii komentorivistä jokaisen osan erilleen, minkä jälkeen se kääntää jokaisen osan heksadesimaalimuotoiseksi merkkijonoksi. Varsinaiseen DigRF-datakehykseen vaaditaan datakuorman pituuden tavuina ilmoittava osio. Datakuorman pituuden kääntäjä pystyy laskemaan helposti aikaisemmin muodostetusta heksadesimaalimerkkijonosta, koska jokainen heksadesimaalimerkki vastaa yhtä tavua. Kun kaikki osiot on käännetty, ne yhdistetään yhdeksi kokonaiseksi merkkijonoksi. Tämän jälkeen kääntäjä tarkistaa, että muodostettu heksadesimaalimerkkijono on pituudeltaan oikea. Renesasin BBIC-emulaattorille vaadittava 32 tavulla jaollinen datakehyksen pituus saavutetaan tarvittaessa lisäämällä 0-tavuja merkkijonon loppuun. Kääntäjän tarkistettua merkkijonon pituuden oikeellisuuden se lisää käsitellyn komentorivin aikaisempien kanssa samaan merkkijonoon. Kuvassa 8 on esitetty binäärikääntäjän toiminta vuokaaviona.



KUVA 8. Binäärikääntäjän vuokaavio

Käytyään läpi kaikki komentorivit kääntäjä on tuottanut yhden heksadesimaalimerkkijonon, joka koostuu kaikista binäärikääntäjälle syötetyistä komentoriveistä. Tämän jälkeen kääntäjä muodostaa heksadesimaalimerkkijonosta binäärimuotoisen taulukon. Kun koko heksadesimaalimerkkijono on käännetty binääridataksi, se kirjoitetaan väliaikaiseen binääritiedostoon. Väliaikainen tiedosto lähetetään BBIC-emulaattorille ohjauksen myöhemmässä vaiheessa.

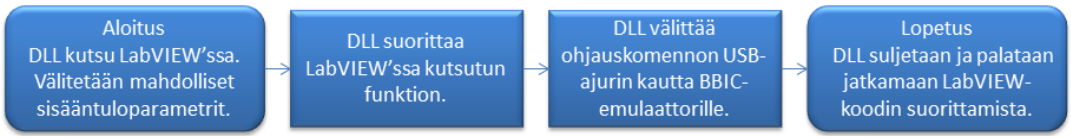
4.2 Ohjauksien toteutus

Työssä tehdyt ohjaukset voidaan jakaa kahteen eri ryhmään, BBIC-emulaattorin ohjauksiin ja RFIC-piirin ohjauksiin. BBIC-emulaattorin ohjaukset sisältävät ainoastaan itse emulaattorilevylle USB-rajapinnan kautta annettavien komentojen toiminnan. RFIC-piirin ohjauksissa on DigRF-kehykseen tarvittava data, joka välitetään piirille BBIC-emulaattorin ohjausten kautta.

4.2.1 BBIC-emulaattorin ohjaukset

Ennen varsinaisten RFIC-piirin ohjausten toteuttamista täytyi tehdä ohjaukset BBIC-emulaattorille. Tietokoneen ja BBIC-emulaattorin välisessä kommunikoinnissa käytettiin apuna olemassa olevaa dynaamista linkkikirjastoa (DLL). Nämä BBIC-emulaattorin ohjausfunktiot muodostavat LabVIEW'lla tehdyn ohjausrajapinnan alimman kerroksen.

Käytetty DLL on Renesasin Ranskan toimipisteessä tehdystä BBIC-emulaattorin ohjausohjelmasta, mutta tämän enempää ohjelmasta ei ollut apua automaattisen testausjärjestelmän suhteen. DLL sisältää kaikki BBIC-emulaattorille tarvittavat ohjausfunktiot. Kun ohjausfunktiota kutsutaan dynaamisesta linkkikirjastosta, se suorittaa itse DLL:n koodissa varsinaisen komennon muodostamisen. Tämän jälkeen dynaaminen linkkikirjasto kutsuu USB-ajuria, jonka kautta se välittää komennot BBIC-emulaattorille. Kuvassa 9 on kuvattu BBIC-emulaattorin ohjauksen toimintaa vuokaavion avulla.



KUVA 9. Vuokaavio BBIC-emulaattorin ohjauksien suorituksesta.

LabVIEW'ssa BBIC-emulaattorin ohjauksien toteutus oli suhteellisen helppoa valmiin dynaamisen linkkikirjaston ansoista. DLL-funktion kutsuminen LabVIEW'ssa onnistuu helposti Call Library Function Node -toiminnon avulla. Call Library Function Node -toimintoon määritetään ohjelmointi vaiheessa käytetyn dynaamisen linkkikirjaston polku, josta sitä kutsutaan. Tämän jälkeen Call Libray Function Node -toiminnon asetuksiin latautuu kaikki DLL:n sisältämät funktiot. Asetuksista valitaan haluttu funktio ja nimetään funktiolle tarvittavat sisääntulo- ja ulostuloparametrit. Parametrien täytyi olla täsmälleen samassa järjestyksessä kuin DLL-funktiossa ja parametrien tyypit täytyi vastata linkkikirjastossa olevia tietotyyppejä. Jotta järjestys ja tyypit pystyttiin

selvittämään, täytyi osata tulkita dynaamisessa linkkikirjastossa käytettyä C++-ohjelmointikieltä.

Pääosin sisääntuloparametreja käytettiin, kun BBIC-emulaattorille välitettiin DigRF-yhteyttä pitkin kirjoitettavat komennot binääritedostona. BBIC-emulaattorin alustuksessa jouduttiin kuitenkin lataamaan emulaattorilevyllä olevan FPGA-piirin laiteohjelmisto binääritiedostona. Joihinkin BBIC-emulaattorin ohjauskomentoihin ei ollut yhtään sisääntuloa, esimerkiksi DigRF-yhteyden avaus- ja sulkemiskomennoissa ja virheenluvussa BBIC-emulaattorilta.

Kaikista BBIC-emulaattorin dynaamisen linkkikirjaston funktioista tehtiin yksittäinen aliohjelma. Aliohjelmia on helppo hyödyntää ohjauksen suuremmassa kokonaisuudessa. Aliohjelmien muodostaminen oli järkevää myös siksi, ettei BBIC-emulaattorin ohjauskomentoja koskaan tarvitse ajaa yksittäin.

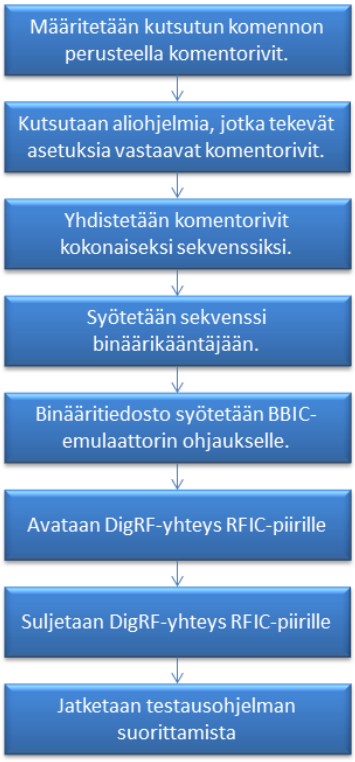
4.2.2 RFIC-piirin ohjaukset

RFIC-piirin ohjaukset ovat toiseksi alimmalla tasolla ohjausrajapinnan toteutuksessa. Tällä tasolla muodostetaan RFIC-piirin ohjauskomennot haluttujen asetusten mukaisesti. Jokaiselle standardille, 2G TX/RX, 3G TX/RX ja LTE TX/RX, täytyi tehdä omat aliohjelmat, joita käytettiin myöhemmin hyödyksi ylemmällä tasolla.

RFIC-piirin ohjausaliohjelmiin sisääntulona välitetään ylemmältä tasolta suoritettava komento ja RFIC-piirin asetukset. Suoritettava komento tarkoittaa RFIC-piirin suorittamaa toimintoa, esimerkiksi 2G TX -lähettimen päälle laittamista tai tehotason vaihtamista. Asetuksissa taas välitetään RFIC-piirille tieto muun muassa käytettävästä taajuudesta, tehotasosta ja kaistanleveydestä.

Kun RFIC-piirin ohjausaliohjelmalle on syötetty haluttu komento, se muodostaa sen perusteella osan tarvittavista komentoriveistä. Asetuksien muuttamisessa komentoriveiksi käytettiin aikaisemmalle BBIC-emulaattorille tehdyn ohjausrajapinnan kanssa samoja aliohjelmia. Kun asetuksista on saatu oikeat komentorivit, yhdistetään kaikki muodostetut komentorivit yhdeksi yhtenäiseksi

sekvenssiksi. Tämän jälkeen komentorivisekvenssi syötetään aikaisemmin kuvattuun binäärikääntäjään, joka kääntää sekvenssin binääritiedostoksi. Binääritiedosto syötetään BBIC-emulaattorin ohjaukselle, joka kirjoittaa sen emulaattorin muistiin FPGA-piirille. Kun koko tiedosto on saatu kirjoitettua BBIC-emulaattorin muistiin, kutsutaan DigRF-yhteyden käynnistävää emulaattoriohjausta. BBIC-emulaattorilta kysellään varmistusta milloin DigRF-komennot on saatu välitettyä RFIC-piirille. Kun varmistus on saatu, DigRF-yhteys suljetaan ja testausohjelmassa edetään seuraaviin vaiheisiin. Kuvassa 10 on selvennetty RFIC-piirin ohjauksen toimintaa vuokaavion avulla.



KUVA 10. Vuokaavio RFIC-piirin ohjauksen toiminnasta.

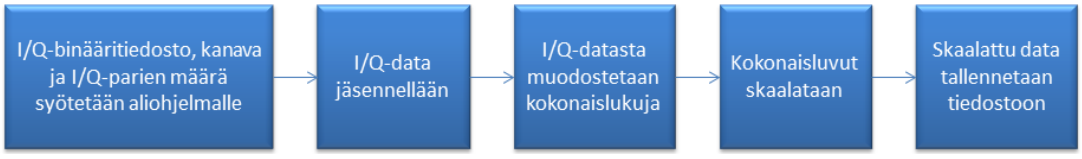
Tämä edellä kuvattu toiminta on samanlainen kaikilla RFIC-piirin ohjausaliohjelmilla. Kun komentosekvenssi on muodostettu, tutkitaan ohjelmassa, löytyykö syötetty I/Q-tiedosto. Mikäli tiedostoa ei löydetä, pyydetään käyttäjää antamaan uusi tiedosto. Ohjelman saatua oikeanlaisen I/Q-tiedoston, suoritetaan komentosekvenssin kirjoitus RFIC-piirille aikaisemmin kuvatulla tavalla. Heti komentosekvenssin kirjoituksen jälkeen RFIC-piirille

kirjoitetaan I/Q-tiedosto. Kirjoitus tapahtuu erilaisella BBIC-emulaattorin ohjauksella kuin ohjaustiedostojen kirjoitus. I/Q-tiedostolle käytettävä ohjausaliohjelma mahdollistaa I/Q-datan toiston RFIC-piirille. Kun I/Q-data on kirjoitettu BBIC-emulaattorin muistiin, suoritetaan DigRF-yhteyden avauskomento. DigRF-yhteys pidetään päällä koko mittauksen ajan, jolloin I/Q-data toistetaan ja RFIC-piiri pystyy lähettämään oikeanlaista signaalia.

4.3 RX I/Q-datan kaappaus

BBIC-emulaattorille tehtiin aliohjelma, joka mahdollisti I/Q-datan kaappaamisen RFIC-piiriltä ja tallentamisen tietokoneelle. Kaapatun I/Q-datan analysoinnista saadaan RX-mittauksen tulokset, joten datan täytyi olla analysointiohjelman tukemassa muodossa. I/Q-datan kaappausaliohjelma vei koko ohjausrajapinnassa eniten aikaa ja siksi se toteutettiin vertailun vuoksi kahdella eri tavalla, LabVIEW'lla ja C++-koodilla.

I/Q-data saadaan BBIC-emulaattorilta binääritedostona, josta saadaan jäsenneltyä halutun kanavan datakehykset erilleen. Binääritiedostossa eri standardien I/Q-data on jaettu omiksi loogisiksi kanaviksi, joista on helppo tunnistaa ja eritellä haluttu I/Q-data. Aliohjelmalle syötetään haluttujen I/Q-dataparien määrä, jolloin ohjelma jäsentelee binääritiedostosta vain tarvittavan määrän dataa. Kun oikeat datakehykset on jäsennelty tiedostosta, käännetään niissä oleva data kokonaisluvuiksi. Kokonaisluvut joudutaan vielä skaalaamaan RX-mittauksissa käytetyn VSA-ohjelman tukemaan muotoon. Skaalattu I/Q-data tallennetaan tekstitiedostoon, joka voidaan ladata analysointiohjelmaan. Edellä kuvattu prosessi esitettynä vuokaaviolla kuvassa 11.



KUVA 11. I/Q-datan kaappaus esitettynä vuokaaviolla

Ensiksi I/Q-datan kaappausaliohjelma toteutettiin LabVIEW'lla. Binääritiedoston jäsentely osoittautui LabVIEW-koodilla liian hitaaksi optimointiyrityksistä huolimatta. Testausjärjestelmälle suurimpana vaatimuksena oli hyvä suoritusnopeus, joten hidasta binääritiedoston jäsentelyä ei hyväksytty. Kaappauksen nopeuttamiseksi kysyttiin Ranskan toimipisteeltä mahdollisuutta tehdä kaappaus sekä kehysrakenteiden purku ja skaalaus BBIC-emulaattorin FPGA-piirillä. FPGA-piirille tehtyyn kaappaukseen vaadittu työmäärä kuitenkin osoittautui liian suureksi, joten Ranskan toimipisteellä toteutettiin 3G-standardille I/Q-datan kaappaus C++-koodilla. C++-koodilla toteutetun kaappauksen nopeus osoittautui noin kymmenen kertaa nopeammaksi, mikä vastaa hyvin automaattisen testausjärjestelmän asettamia vaatimuksia.

Nopeutensa takia Ranskasta saatua C++-koodia alettiin jatkokehittää ja koodiin tehtiin tuki kaikille standardeille. Koodiin tehtiin muutama pieni lisäys toiminnan parantamiseksi, muun muassa I/Q-dataparien määrän valinta. C++-koodista käännettiin DLL, jonka funktioita voitiin helposti kutsua LabVIEW'lla. Ainoastaan tässä osiossa opinnäytetyötä jouduttiin itse käyttämään C++-koodia ohjausrajapinnan toteutuksessa.

4.4 Integrointi testausjärjestelmään

Ohjausrajapinnan integrointi testausjärjestelmään vaati käytännössä muutoksia kolmeen eri kohtaan varsinaisen testausjärjestelmän koodiin. Muutokset tehtiin RFIC-piirin ohjauksen valintaan, TX-testeissä testitapauksen aloitukseen ja RX-testeissä VSA:n suorittamaan analysointiin. Tässä luvussa on selitetty tarkemmin testausjärjestelmään tehdyt muutokset.

4.4.1 Ohjauksen valinta

Varsinaisen testausjärjestelmän koodiin, jossa tehdään valinta RFIC-piirin ohjausrajapinnasta, lisättiin uusi aliohjelma. Aliohjelma on BBIC-emulaattorin ohjausrajapinnan ylin kerros ja se lukee käyttäjän määrittämiä komentoja sekä kutsuu komentojen mukaisia BBIC-emulaattorin ohjauksia.

Tätä ohjausrajapinnan osiota kutsutaan aina ensimmäisenä uutta mittausta aloitettaessa ja se on varsinainen rajapinta BBIC-emulaattorin ohjauksien ja

käyttäjän välillä. Käyttäjän antamien komentojen perusteella aliohjelma kutsuu eri BBIC-emulaattorin ohjausten suorittamisia.

Tämä osio BBIC-emulaattorin ohjausrajapinnasta lukee ensiksi käyttäjän asettaman komennon ja sen jälkeen käyttäjän määrittämät asetukset. Esimerkiksi käyttäjä määrittää ensiksi laiteohjelmiston latauskomennon BBIC-emulaattorille. Komennon luettuaan aliohjelma lukee käyttäjän määrittämän laiteohjelmistotiedoston sijainnin ja sen jälkeen välittää sen alemmille ohjausrajapinnan kerroksille.

4.4.2 TX-testitapaukset

TX-mittauksissa täytyi tehdä pieniä muutoksia mittauksen aloitukseen ja lopetukseen. TX-mittauksissa RFIC-piiri tarvitsee BBIC-emulaattorilta I/Q-dataa, jota RFIC-piiri käyttää signaalina.

Mittauksen aloitukseen tehtiin I/Q-tiedoston lataaminen BBIC-emulaattorille. Mittauksissa I/Q-data ladataan BBIC-emulaattorille toistettavaksi juuri ennen varsinaisen mittauksen aloittamista. Toteutuksessa käytettiin ainoastaan alimman tason aliohjelmia eli BBIC-emulaattorin ohjauksia. Käyttäjän antama I/Q-tiedoston polku luettiin asetuksista ja syötettiin aliohjelmaan, joka kirjoittaa tiedoston BBIC-emulaattorin muistiin. Tämän jälkeen suoritettiin aliohjelma, joka avaa DigRF-yhteyden. Yhteyden ollessa auki BBIC-emulaattori toistaa I/Q-dataa RFIC-piirille yhtäjaksoisesti.

Mittauksen lopetuksessa ainoana muutoksena oli sulkea päällä oleva DigRF-yhteys. Yhteyden sulkeminen lopettaa I/Q-datan toistamisen BBIC-emulaattorilla ja poistaa binäärimuodossa ladatun I/Q-datan emulaattorin muistista. Tämän jälkeen testausohjelman suorittamista jatketaan normaalisti.

4.4.3 RX-testitapaukset

RX-mittauksissa muutokset olivat isompia kuin TX-mittauksiin tehdyt muutokset. RX-mittauksissa tulokset analysoidaan VSA-ohjelmalla RFIC-piiriltä kaapatusta I/Q-datasta, joka on muutettu VSA-ohjelman tukemaan tiedostomuotoon.

Kaappausta varten ohjausrajapintaan jouduttiin toteuttamaan uusi aliohjelma, josta selitettiin tarkemmin luvussa 4.3.

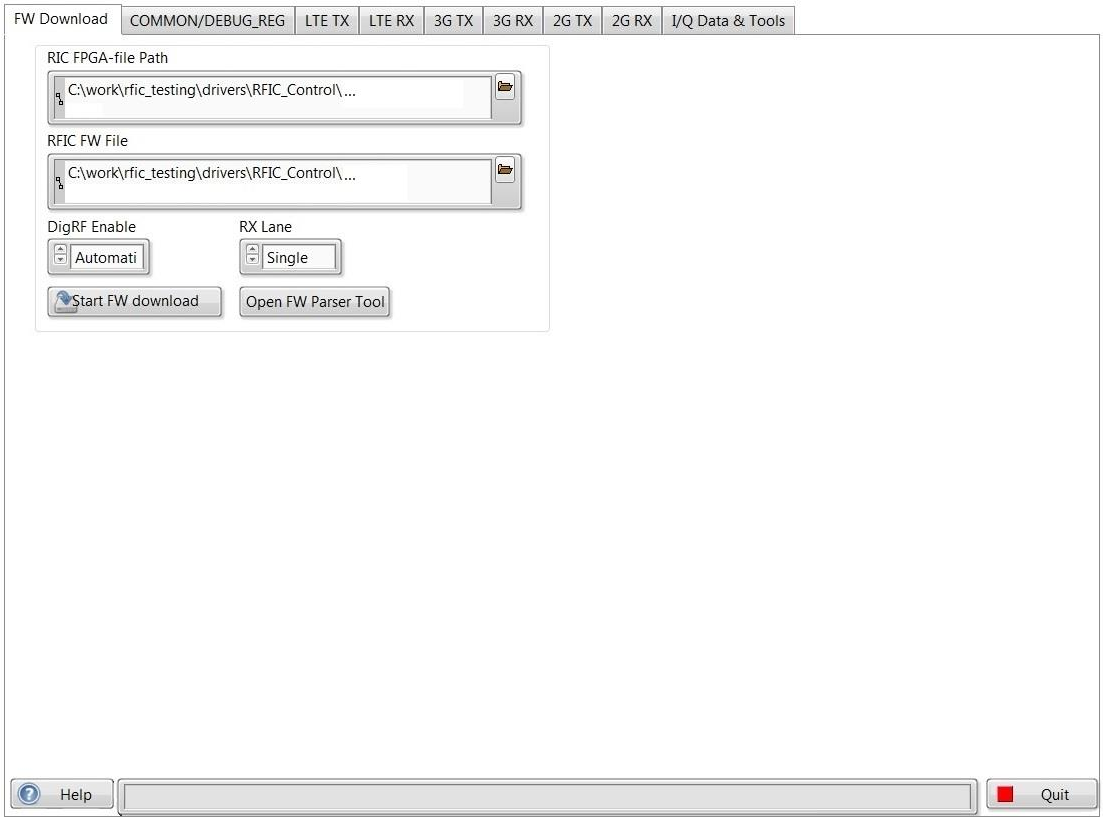
Testausjärjestelmän RX-mittauksen koodissa jouduttiin tutkimaan käyttäjän määrittelemästä näytemäärästä ja näytteenottotaajuudesta, kuinka pitkään BBIC-emulaattorin RX-vastaanotinta joudutaan pitämään päällä, jotta saadaan tarpeeksi näytteitä tallennettua. Oikea aika oli tärkeää saada, ettei mittausaikaa kulu turhaan. Näytteet tallennetaan I/Q-tiedostoon, joka syötetään mittauksen jälkeen C++-koodilla tehdylle aliohjelmalle jäsenneltäväksi ja skaalattavaksi. Kun I/Q-tiedosto on oikeassa muodossa, voidaan se analysoida VSA-ohjelmalla. Analysoinnin jälkeen tulokset tallentuvat automaattisen testausjärjestelmän muistiin ja aloitetaan uusi mittaus tai lopetetaan mittaussekvenssi ja näytetään tulokset käyttäjälle.

4.5 Itsenäinen ohjauskäyttöliittymä

Toisena tavoitteena opinnäytetyöhön oli asetettu itsenäisen ohjauskäyttöliittymän toteuttaminen. Itsenäisen ohjauskäyttöliittymän tarkoituksena oli pystyä ohjaamaan RFIC-piiriä manuaalisesti. Itsenäinen ohjauskäyttöliittymä oli hyödyllinen testeissä, joissa ei tarvinut käyttää automatiikkaa sekä RFIC-piirin ongelmatilanteiden tulkinnassa.

Itsenäinen ohjauskäyttöliittymä on kokonaan riippumaton automaattiseen testausjärjestelmään integroidusta ohjausrajapinnasta. Ohjauskäyttöliittymä kuitenkin käyttää alimmille tasoille tehtyjä ohjauksia eli RFIC-piirin ohjauksia ja BBIC-emulaattorin ohjauksia. Käytännössä itsenäisestä ohjauskäyttöliittymässä on tuotu käyttäjän ulottuville automaattiseen testausjärjestelmään integroidun ohjauksien valinta -aliohjelman ominaisuudet. Ohjauskäyttöliittymä ei lue asetuksia mistään valmiista datasta, vaan käyttäjä joutuu itse manuaalisesta valitsemaan yhden toiminnon kerrallaan, esimerkiksi käyttäjä valitsee laiteohjelmistotiedoston polun ja painiketta painamalla lataa sen RFIC-piirille. Itsenäisestä ohjauskäyttöliittymästä julkaistiin oma asennuspakettinsa, koska sillä ei ole mitään kytköksiä varsinaiseen automaattiseen testausjärjestelmään.

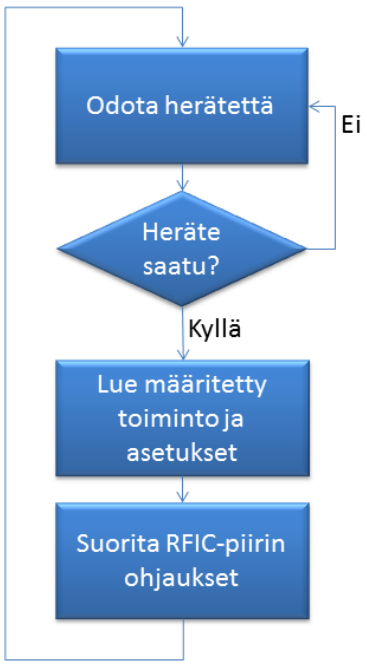
Ennen varsinaisten toimintojen lisäämistä LabVIEW’n lohkokaavioon suunniteltiin alustava käyttöliittymän etupaneeli. Käyttöliittymään tuli standardien asetukset eri välilehdille, mikä havainnollistaa eri toiminnot käyttäjälle hyvin. Kuvassa 12 on osa lopullisesta käyttöliittymään toteutetusta etupaneelista.



KUVA 12. Esimerkki itsenäisen ohjauskäyttöliittymän etupaneelista

Suurin osa itsenäisen ohjauskäyttöliittymän toiminnoista pystyttiin toteuttamaan samalla tavalla kuin automaattiseen testausjärjestelmään integroidussa ohjausaliohjelmassa. Jotkin toiminnot, kuten I/Q-datan kaappaus, jouduttiin toteuttamaan eri tavalla kuin testausjärjestelmään integroidessa. Itsenäisessä ohjauskäyttöliittymässä on myös joitakin toimintoja, joita ei tarvittu toteuttaa varsinaiseen automaattiseen testausjärjestelmään. Tällainen toiminto on esimerkiksi RFIC-piirin tilan luku. Toiminnallisesti ohjauskäyttöliittymä vaatii aina käyttäjän antamia herätteitä etupaneelilta ja tekee sen mukaiset ohjaukset RFIC-piirille. Kun RFIC-piirin ohjaukset on suoritettu, ohjauskäyttöliittymä palaa

odottamaan käyttäjän seuraavaa herätettä etupaneelista. Kuvassa 13 on esitetty yksinkertaisen vuokaavion avulla itsenäisen ohjauskäyttöliittymän perustoiminta.



KUVA 13. Vuokaavio ohjauskäyttöliittymän toiminnasta

Varsinaisen ohjauskäyttöliittymän saa suljettua kuvassa 12 näkyvästä Quit-painikkeesta aina, kun ohjauskäyttöliittymä odottaa käyttäjän herätettä. Suljettaessa ohjauskäyttöliittymä ei suorita minkäänlaisia ojauksia RFIC-piirille.

5 TESTAUS

Vaikka varsinaista toiminnallisuuden testausta tehtiin koko ohjausrajapinnan kehityksen ajan, niin testaus voidaan silti jakaa useampaan eri kokonaisuuteen. Toteutusvaiheessa tehdyt testaukset eivät kuitenkaan olleet niin laaja-alaisia kuin loppuvaiheessa tehty täysimittainen testaus. Itsenäistä ohjauskäyttöliittymää pystyttiin hyödyntämään erittäin hyvin ongelmatilanteiden syntyessä. Se mahdollisti hyvin spesifisen testauksen ja näin ollen ongelman helpon paikantamisen.

5.1 Ohjausten testaus

Toteutettujen ohjauksien toiminta testattiin aina niiden toteuttamisen jälkeen nopeasti. Ensimmäinen viite ohjauksen toiminnasta saatiin virtalähteestä nähtävän virrankulutuksen muuttumisesta. Esimerkiksi laiteohjelmiston latautuessa onnistuneesti RFIC-piiri alkoi kuluttaa hieman enemmän virtaa kuin ilman laiteohjelmistoa.

RFIC-piirin TX-toiminnallisuutta testattaessa apuna käytettiin VSA:ta. RFIC-piirin TX-lähettimen ohjauksia testatessa tarkistettiin, että VSA:lle ilmestyi signaalipiikki asetetun taajuuden kohdalle. Pelkkä signaalipiikki näkyy VSA:lla silloin, kun TX-lähetin on päällä ja RFIC-piirille ei ole syötetty oikeaa signaalia muodostavaa I/Q-dataa toistettavaksi. RFIC-piirin lähettäessä I/Q-datasta muodostettua signaalia oikein, VSA:lla näkyy kullekin eri standardille ominainen signaalin muoto. VSA:lta pystytään siis toteamaan, että TX-lähettimen päälle laitto ja I/Q-datan syöttäminen RFIC-piirille toimii oikein.

RFIC-piirin RX-toiminnallisuus, ennen I/Q-datan kaappauksen tekoa, todennettiin virrankulutuksen ja BBIC-emulaattorilta saadun binääritiedoston perusteella. RX-vastaanottimen päällä ollessa RFIC-piiri kuluttaa hieman enemmän virtaa kuin lepotilassa ollessa. BBIC-emulaattorilta saadusta binääritiedostosta pystyttiin tutkimaan saadaanko tiedoston mukana haluttuja I/Q-datakehyksiä, joilla on oma tietynlainen rakenteensa.

5.2 I/Q-datan kaappauksen testaus

I/Q-datan kaappausaliohjelman ja RFIC-piirin RX-vastaanottimen testaus toteutettiin osaksi limittäin, koska ilman varmasti toimivaa I/Q-datan kaappausaliohjelmaa ei voitu olla varmoja RFIC-piirin lähettämän I/Q-datan oikeellisuudesta. I/Q-datan kaappauksesta testattiin muunnettujen I/Q-dataparien oikeellisuutta ja aliohjelman suoritusnopeutta.

VSA-ohjelmaa käytettiin varmistamaan BBIC-emulaattorilta saadun I/Q-datan oikeellisuus. Kun binääritiedostossa oleva I/Q-data oli käännetty VSA-ohjelmalle sopivaan muotoon, käännetty tiedosto syötettiin analysointiohjelmaan ja tutkittiin, onko signaali oikean muotoinen. Tarkasti signaalin oikeellisuus saatiin tietää vasta täysimittaisessa testauksessa, jossa ajettiin automaattisella testausjärjestelmällä varsinaisia testimittauksia läpi.

Toteutusvaiheessa testattiin ja vertailtiin myös LabVIEW'lla ja C++-koodilla toteutettujen aliohjelmien suoritusnopeutta. Tätä varten tehtiin LabVIEW'lla testipenkki, johon pystyi vaihtamaan I/Q-datan kaappauksen suorittaman aliohjelman. Testipenkki laski kaappauksen keston, jota verrattiin aliohjelmien kesken. Optimointiyrityksistä huolimatta LabVIEW'lla toteutettu aliohjelma ei yltänyt lähellekään C++-koodilla toteutetun aliohjelman suoritusaikoja. Näin ollen testauksen tuloksena päädyttiin käyttämään C++-koodilla toteutettua aliohjelmaa automaattiseen testausjärjestelmään integroidussa ohjausrajapinnassa.

5.3 Täysimittainen testaus

Lopulliseen testaukseen ohjausrajapinta joutui, kun sitä alettiin käyttää oikeassa mittausympäristössä, jossa käytettiin kehitettyä automaattista testausjärjestelmää ja oikeita testimittauksia. Tässä testausvaiheessa nähtiin, toimivatko kaikki ohjausrajapinnan osat kunnolla yhteen ja oliko ohjausrajapinnan integrointi onnistunut.

Testatessa huomattiin, että osa BBIC-emulaattorin ohjausrajapinnan aliohjelmista vaati pientä hienosäätöä stabiilisuusongelmien vuoksi. Lopullisessa testauksessa mittauksia ajettiin suuria määriä yhtäjaksoisesti,

jolloin osa ohjausrajapinnan aliohjelmista saattoi välillä jumittua. Aikaisemmin näitä ongelmia ei tullut ilmi, koska testausta tehtiin pieninä kokonaisuuksina ja testimittausten määrät eivät olleet suuria.

RFIC-piirin ohjauksia, erityisesti I/Q-datan toisto TX-puolella ja kaappaus RX-puolelle, pystyttiin testaamaan tarkasti oikeista mittauksista saatujen tulosten ansiosta. Tuloksia pystyttiin vertaamaan aikaisemmin käytetyllä BBIC-emulaattorilla saatuihin tuloksiin ja tästä päättelemään, ovatko uudella BBIC-emulaattorilla mitatut tulokset sallittujen rajojen sisässä.

6 YHTEENVETO

Tämän opinnäytetyön tarkoituksena oli toteuttaa Renesasin kehittämälle BBIC-emulaattorille ohjausrajapinta kehitteillä olevaan automaattiseen RFIC:n testausjärjestelmään. Sen tuli vastata testausjärjestelmän vaatimaa suorituskykyä ja nopeutta. Lisätavoitteena oli tehdä BBIC-emulaattorille itsenäinen ohjauskäyttöliittymä, joka ei ollut mitenkään riippuvainen automaattisesta testausjärjestelmästä.

Työn ohjelmointiympäristönä toimi LabVIEW. LabVIEW oli luonnollinen valinta, koska kehitteillä oleva automaattinen RFIC-piirin testausjärjestelmää tehtiin LabVIEW'lla.

Työn tavoitteet saavutettiin ja BBIC-emulaattorin ohjausrajapinta saatiin integroitua onnistuneesti automaattiseen testausjärjestelmään. Tämä mahdollistaa testauksen monilla eri paikoilla, koska käyttöön saadaan useampi BBIC-emulaattori. Myös Renesasin kehittämä BBIC-emulaattori todettiin testejä ajettaessa hieman nopeammaksi kuin laitevalmistajalta hankittu BBIC-emulaattori.

Ensimmäisen tavoitteen lisäksi opinnäytetyön toinen tavoite pystyttiin toteuttamaan loppuun asti. Itsenäinen ohjauskäyttöliittymä saatiin toteutettua toimintakelpoiseksi ja sitä voidaan hyödyntää pienemmissä testeissä ja RFIC-piirin ongelmatilanteiden tulkitsemisessa.

LÄHTEET

1. What Is LabVIEW. 2010. National Instruments. Saatavissa: http://www.ni.com/labview/whatis/. Hakupäivä 11.3.2013.

2. Nasser, Kehtarnavaz – Namjin, Kim 2005. Digital Signal Processing System-Level Design Using LabVIEW. Elsevier.

3. DigRF(SM) Specification. 2013. MIPI Alliance. Saatavissa: http://www.mipi.org/specifications/digrfsm-specifications. Hakupäivä 21.3.2013.

4. DigRFSM v4 v1.10 Specification Data Sheet. 2012. MIPI Alliance. Saatavissa: http://mipi.cloudfour.com/sites/default/files/DigRF%20data%20sheet%20final%20v1%20011212.pdf Hakupäivä: 19.4.2013

5. N5182A MXG RF Vector Signal Generator. 2013. Agilent Technologies. Saatavissa: http://www.home.agilent.com/en/pd-797248-pn-N5182A/mxg-rf-vector-signal-generator?nid=-536906709.536910812&cc=FI&lc=fin. Hakupäivä 25.3.2013.

6. Agilent Vector Signal Analysis Basics. Agilent Technologies. 2012. Saatavissa: http://cp.literature.agilent.com/litweb/pdf/5990-7451EN.pdf. Hakupäivä: 24.3.2013.